Journal of Uncertain Systems
Vol.11, No.2, pp.84-103, 2017
Online at: www.jus.org.uk

Model-Order Reduction Using Interval Constraint Solving Techniques

Leobardo Valera, Martine Ceberio*
Computer Science Department, University of Texas at El Paso, El Paso, TX 79968, USA

Received 4 October 2015; Revised 10 December 2015

Abstract

Many dynamical phenomena can be modeled as ordinary or partial differential equations. A way to find solutions of such equations is to discretize them and to solve the corresponding (possibly) nonlinear and large systems of equations; see [19].

Solving a large nonlinear system of equations is very computationally complex due to several numerical issues, such as high linear-algebra cost and large memory requirements. Model-Order Reduction (MOR) has been proposed as a way to overcome the issues associated with large dimensions, the most used approach for doing so being Proper Orthogonal Decomposition (POD); see [25]. The key idea of POD is to reduce a large number of interdependent variables (snapshots) of the system to a much smaller number of uncorrelated variables while retaining as much as possible of the variation in the original variables.

In this work, we show how intervals and constraint solving techniques (see [8, 18, 22]) can be used to compute all the snapshots at once and propose a new model-order reduction technique (that we call I-POD). This new process gives us two advantages over the traditional POD method: 1. handling uncertainty in some parameters or inputs; 2. reducing the snapshots computational cost. We illustrate our proposed method, I-POD, on several nonlinear problems.

Keywords: model-order reduction, proper orthogonal decomposition, large nonlinear systems of equations, interval constraint solving techniques

1 Introduction

Many real life phenomena or situations can be represented by mathematic models. Because of the dynamical nature of these phenomena, the associated models could be, for instance, partial differential equations (PDEs). Such differential equations arise in many engineering problems describing phenomena such as the distribution of heat in a given rod or plate over time (heat equation), the description of waves like those of vibrating strings, and sound and water waves (wave equation), gas dynamics and traffic flow (Burgers' equation); see [26, 27].

A way to find an approximation of the solution of differential equations, either ordinary or partial, is to discretize the domain of the solution and to form a system of algebraic equations: the resulting system of equations can be linear or nonlinear, depending on the nature of the PDE.

In order to obtain a good accuracy in the approximation of the sought solution, the domain has to be discretized in many elements and nodes, leading to a large system of equations. Now common issues in solving such systems are: 1. not knowing about the existence and/or uniqueness of the solution of the system of equations, 2. storage, 3. high computational cost, and 4. rounding errors.

To overcome these issues, several techniques have been developed to allow to solve a smaller system with similar features instead of the large original one, hence addressing storage and computation issues. This is done by finding a subspace where an acceptable approximation of the solution of the system of equations lie. This process of identifying such subspace and reducing a large problem to a smaller one is known as **Model-Order Reduction (MOR)**.

Proper Orthogonal Decomposition (POD) is a broadly used and effective method to identify a reduced subspace and reduce the original large problem to a much smaller one. This method is based on

*Corresponding author.
Emails: lvalera@utep.edu (L. Valera), mceberio@utep.edu (M. Ceberio).

Principal Component Analysis (PCA) [17, 21, 24]. The idea behind POD consists in the following: solve the large nonlinear system for a certain number of input parameter values so as to generate a broad set of behaviors of the system at hand; extract from these runs meaningful features; use these as a reduced basis; and solve the original system on the reduced basis only, hence yielding much faster performance and a good quality approximation (if the reduced basis was designed properly).

Although POD is one of the most popular approaches to reducing the order of large systems of equations, it presents several disadvantages, the main drawback being that it requires a series of offline computations in order to form the matrix of snapshots. The quality of the resulting reduced basis heavily depends on the choice of parameters and inputs during the offline phase, and on the accuracy of these over which the snapshots are computed.

In this work, we propose an alternative method for reducing the order of large systems of equations. It explores the idea of computing snapshots as a result of interval computations that we then sample: this leads us to one – interval – solving process as opposed to solving the large system as many times as we need snapshots. This led to shaping our proposed method, which we called Interval-POD (I-POD). Let us note that I-POD has advantages beyond the mere computations of snapshots: if POD can handle intervals, it can therefore handle uncertainty as well. This would allow models to factor in uncertainty while still being processed as a reduced-order model. Our preliminary experiments on a series of nonlinear problems show promise.

2 Background

Let us start by recalling the type of problems that we are attempting to solve. Many real-life phenomena are modeled and result in very large (most likely) nonlinear systems of equations that need to be solved. Solving these problems boils down to finding the zeroes of large-dimensional functions. Traditionally, finding zeroes of functions is achieved via the use of Newton methods.

In this section, we review basic notions about the components that motivate and make up our approach. Namely, we start by recalling the Newton's approach, which motivates the need to Model-Order Reduction. We then go over the MOR concept. Finally, we review interval computations and interval constraint solving techniques, as they are essential to our proposed I-POD.

2.1 The Newton Method

The Newton method is an iterative procedure that finds the zeroes of continuously differentiable functions $F : \mathbb{R}^n \to \mathbb{R}^n$. The formulation of the method is given by:

$$J_F(x_n)(x_{n+1} - x_n) = -F(x_n) \tag{1}$$

where $J_F(x_n)$ is the $n \times n$ Jacobian matrix of F.

If F is twice differentiable and the Hessian $\nabla^2 F(x)$ is Lipschitz continuous in a neighborhood of a solution x^* then:

1. if the initial point x_0 is sufficiently close to x^*, the sequence of iterations converges to x^*; and
2. the rate of convergence of $\{x_k\}$ is quadratic.

The Newton method is outlined in Table 1:

Table 1: Newton method outline

Given an initial point x_0
`for i=1 until convergence`
Compute $F = F(x_0)$ and $J = J_F(x_0)$
Solve the linear system of equations: $J\Delta x = -F$
Compute: $x_{i+1} = x_i + \Delta x$
`end for`

The Newton method converges if certain conditions are satisfied; for example, if a stationary initial point is chosen or if the approach outlined above enters in a cycle, the Newton method will not converge. Also, if the Jacobian matrix is singular or if any of its entries is discontinuous at the root, the convergence may fail. If the Jacobian is singular at the root of the function or the Hessian is not defined at it, the process may converge but not in q-quadratic order.

In addition to the above limitations of the Newton's approach, let us recall that the systems that are being considered for solution are of very large size. Not meeting a q-quadratic order of convergence is much more critical on such large spaces than it would be on smaller spaces.

To overcome all issues above mentioned, the solution is sought on a subspace where the convergence conditions are met, hence Model-Order Reduction.

2.2 Model-Order Reduction

The main idea of the concept of Model Order Reduction (MOR) is as follows:

Let $T : V \to V$ be a bijective linear transformation. Then for every $b \in V$, there exists a unique $x \in V$ such that $T(x) = b$. Every linear transformation has a matrix representation [12]. In this case, let us call A the matrix representation of T. Thus, finding x such that $T(x) = b$ is equivalent to solving the linear system:

$$Ax = b. \tag{2}$$

If the dimension of V is n, then (2) is a $n \times n$ linear system of n equations and n unknowns.

We can assure that there exists W, a subspace of V, whose dimension is $k \ll n$ and such that $x \in W$. This is true because, in particular, the subspace spanned by $\{x\}$ is a subspace of V, which contains x and whose dimension is $1 \ll n$.

Since W is a subspace of V, there exists a base $B = \{w_1, w_2, \ldots, w_k\}$ such that every element $w \in W$ can be expressed as a linear combination of the elements of B [5]. In particular, if $w = x$,

$$x = \sum_{i=1}^{k} y_i w_i. \tag{3}$$

Since every base uniquely determines a subspace of V, we can, without loss generality, speak about subspace W and its base without difference. By writing (3) in matrix form, we obtain

$$Wy = x. \tag{4}$$

After substituting (4) in (2), we have:

$$(AW)y = b, \tag{5}$$

that can be solved using the normal equation [3]

$$(AW)^T(AW)y = (AW)^T b, \tag{6}$$

which is itself a $k \times k$ linear system of equations. After we identify y, we can use (4) to find x.

Once the subspace is found, the approximation of the solution is given as the projection of it on the subspace obtained. This method truncates the solution of the original system to an appropriate basis. Let us illustrate this method by considering a basis transformation T that maps the original n-dimensional state space x into a vector that we will denote by

$$T(x) = \begin{pmatrix} T_1(x) \\ T_2(x) \end{pmatrix} = \begin{pmatrix} \hat{x} \\ \tilde{x} \end{pmatrix}$$

where $\hat{x}$ is k-dimensional. Suppose that T has at least a right-inverse. Let us denote $S = T^{-1}$ then S can be written as

$$S = (S_1 \;\; S_2)$$

and

$$\begin{aligned} I &= \begin{pmatrix} T_1 \\ T_2 \end{pmatrix} (S_1 \;\; S_2) \\ &= \begin{pmatrix} T_1S_1 & T_1S_2 \\ T_2S_1 & T_2S_2 \end{pmatrix} \qquad (7) \\ &= \begin{pmatrix} I_k & 0 \\ 0 & I_{n-k} \end{pmatrix}. \qquad (8) \end{aligned}$$

Since $T_1S_1 = I_k$, we have $\Pi = S_1T_1$ is an oblique projection along the kernel of T_1 onto the k-dimensional subspace that is spanned by the columns of the matrix S_1.

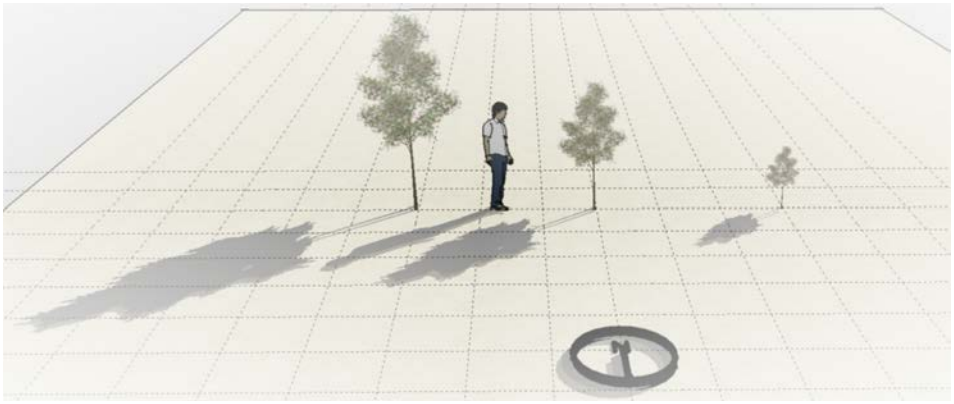


Figure 1: An oblique projection can be see as the shadow cast by objects on the ground when the sun is not directly vertical. Image taken from the site: http://www.schoolkitchengarden.com.au/design-your-garden/

Let

$$\begin{aligned} \frac{dx}{dt} &= f(x,u), \\ y &= g(x,u), \\ x(t_0) &= x_0 \end{aligned} \qquad (9)$$

be the dynamical system, where u is the input of the system, y is the output, x the so-called *state variable*. If we substitute the projection into the dynamical system 9, we obtain

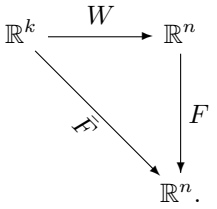
$$\begin{aligned} \frac{d\hat{x}}{dt} &= T_1\hat{f}(S_1\hat{x} + S_2\tilde{x}, u), \\ y &= \hat{g}(S_1\hat{x} + S_2\tilde{x}, u). \end{aligned} \qquad (10)$$

The approximation occurs when we delete the terms involving $\tilde{x}$

$$\begin{aligned} \frac{d\hat{x}}{dt} &= T_1\hat{f}(S_1\hat{x}, u), \\ y &= \hat{g}(S_1\hat{x}, u). \end{aligned} \qquad (11)$$

In order to obtain a good approximation to the original system, the term $S_2\tilde{x}$ must be sufficiently small.

In the nonlinear case, $F(x) = 0$, the variable x is substituted by $x = Wy$. The nonlinear system of equations becomes the overdetermined nonlinear system: $\bar{F}(y) = (F \circ W)(y) = F(Wy) = 0$.



The Jacobian matrix of $\bar{F}$ is defined as:

$$J_{\bar{F}}(y) = J_F(Wy)W. \qquad (12)$$

After using (12), the algorithm shown in Table 1 becomes the Gauss-Newton algorithm shown in Table 2.

Table 2: Gauss-Newton method outline

Given an initial point $y_0 \in \mathbb{R}^k$
`for i=1 until convergence`
Compute $\bar{F} = F(Wy_0)$ and $J_{\bar{F}} = J_F(Wy_0)W$
Solve the linear system of equations: $(J_{\bar{F}}W\Delta)y = -J_{\bar{F}}$
Compute: $y_{i+1} = y_i + \Delta y$
`end for`
$x = Wy^*$

2.3 Interval Constraint Solving Techniques (ICST)

The method that we are proposing in this article consists in expanding POD-based MOR techniques to interval computations (as we will describe in Section 4). Indeed, we aim to group the (possibly many) computational processes over the reals required to generate all snapshots into one single computational process over intervals that encompasses all computations over the reals.

In this subsection, we give a brief overview of interval computations and how to solve systems of equations that involve intervals; for more details about the field, please see [22].

2.3.1 Computations with Intervals

Let us start by pointing that in what follows, when mentioning intervals, we actually mean *closed intervals*. In addition, for simplicity, when we talk about intervals, we will talk about real-value-bounded intervals (not just floating-point-bounded intervals as is commonly the case when implemented on a computer). So in this work, an interval X is defined as follows:

$$X = [\underline{X},\ \overline{X}] = \{x \in \mathbb{R} : \underline{X} \leq x \leq \overline{X}\}. \tag{13}$$

Operations on intervals are simply defined as follows: Since $x \in X$ means that $\underline{X} \leq x \leq \overline{X}$, and $y \in Y$ means that $\underline{Y} \leq y \leq \overline{Y}$ the followings operations are defined based on its infimum and supremum:

$$\textbf{Addition:}\ \ X + Y = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}] \tag{14}$$

$$\textbf{Substraction:}\ \ X - Y = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}] \tag{15}$$

$$\textbf{Multiplication:}\ \ X \cdot Y = [\min S, \max S],\ \text{where } S = \{\underline{X}\underline{Y}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{X}\overline{Y}\} \tag{16}$$

As we observe above, combining intervals with addition, subtraction, and multiplication, always results in one interval. However, it is not always the case without extra care. For instance, the division of an interval by another one that contains 0 should result in two disjunct intervals. To avoid such cases with compromise the nature of traditional interval computations (according to which combining intervals should result in an interval), we generalize the combination of two intervals as follows:

$$\forall X, Y \text{ intervals},\ \ X \diamond Y = \square\{x \diamond y,\ \text{where } x \in X \text{ and } y \in Y\} \tag{17}$$

where $\diamond$ stands for any arithmetic operator, including division, and $\square$ represents the hull operator.

More generally, when carrying out more general computations involving intervals, e.g., computing the interval value of a given function $f : \mathbb{R}^n \to \mathbb{R}$ on interval parameters (or a mix of interval and real-valued parameters), we have the following property:

$$f(X_1, \cdots, X_n) \subseteq \square\{f(x_1, \cdots, x_n),\ \text{where } x_1 \in X_1, \ldots, x_n \in X_n\} \tag{18}$$

where $f(X_1, \cdots, X_n)$ represents the range of function f over the domain $X_1 \times \cdots \times X_n$ and $\square\{f(x_1, \cdots, x_n),\ \text{where } x_1 \in X_1, \ldots, x_n \in X_n\}$ represents the smallest closed interval enclosing this range.

Computing the exact range of f over intervals is therefore a very hard problem and instead, we approximate the range of f over domains using what we call an interval extension of f, which is in fact a surrogate interval function F.

Interval extensions of a given function f have to satisfy the following (very lose) property:

$$f(X_1, \cdots, X_n) \subseteq F(X_1, \cdots, X_n) \tag{19}$$

which to some extent would allow F to be the function that maps any input to the interval $[-\infty, +\infty]$. More pragmatically, the aim is to identify a function F that does not dramatically overestimate the range of our original function f (the closer to the range the better of course, but cost of achieving better range is also an issue). Many interval extensions exists. The most common one is the so-called natural extension, which is a simple interval extension of the syntactical expression of f: arithmetic operations are evaluated using interval rules as shown above, and any other single operator – e.g., power – has its own interval extension; see [22] for more details. Other extensions include Trombettoni et Al.'s occurrence grouping approach [1]. In this work, we use interval computations provided in RealPaver [8] and the natural extensions this software provides.

2.3.2 How to Solve Nonlinear Equations with Intervals?

The premise of our approach is that we will replace several real-valued computational processes by one interval-based computational process by abstracting one real-valued parameter into an interval parameter. Each process (real-valued or interval) consists in solving a (most likely) nonlinear system of equations. In this subsection, we give the reader an overview of the way we proceed to solve a nonlinear system of equations that involves intervals.

We choose to solve nonlinear equations using interval constraint solving techniques. Constraint solving techniques allow to solve systems of constraints. Generally speaking, a constraint describes a relationship that its variables need to satisfy. A solution of a constraint is an assignment of values to the variables of the given constraint such that the relationship is satisfied.

In our case, each of our nonlinear equations $f_i(x_1, \cdots, x_n) = 0$ is a constraint: it establishes a relationship that the values of the variables should satisfy, in this case so that $f_i(x_1, \cdots, x_n)$ be equal to 0. Our system of nonlinear equations is therefore a system of constraints and our goal is to find values of the variables of this system that are such that: $\forall i,\ f_i(x_1, \cdots, x_n) = 0$.

Constraint solving techniques allow us to identify such values of the parameters that satisfy the constraints. Interval constraint solving techniques [20, 13] produce a solution set (set of the solutions of the constraint system) that is interval in nature (this is what you will see in the graphs plotting our experimental results in Section 5): it is a set of multi-dimensional intervals (or boxes whose dimension is n, the number of variables) that is guaranteed to contain all the solutions of the constraint problem (in our case, of the nonlinear system of equations).

The guarantee of completeness provided by interval constraint solving techniques comes from the underlying solving mode: a branch-and-bound [15] (or branch-and-prune for faster convergence [4]) approach that uses the whole search space as a starting point and successively assess the likeliness of finding solutions in the given domain (via interval computations) and possibly (if Branch and Prune) reduce it, and discard domains that are guaranteed not to contain any solution. *Note: while Branch-and-Bound algorithms only assess domains for likeliness of containing a solution (it is a keep or discard approach), Branch-and-Prune algorithms first use the constraints to reduce the domains to consistent domains (using appropriate consistency techniques based on interval computations) and the outcome (empty domain or not, small enough or not to be called a solution) decides whether to continue exploring the domain or not.*

For instance, if on a given domain $D \subset \mathbb{R}$, any of the f_i is such that $0 \notin F_i(D)$, where F_i is an interval extension of f_i, then we can conclude that there is no zero of our system of equations in D and discard it altogether. In Table 3, we outline the generic Branch-and- Bound approach, which is the underlying principle of search in interval constraint solving techniques, and allows to guarantee completeness of the search.

Using interval computations carries a lot of advantages, one of which being that the search can be guaranteed to be complete and that since intervals are used (interval computations to assess whether a domain is a viable option of not), uncertainty can easily be added and seamlessly handled. This however comes at a cost: interval solving processes are usually more computationally taxing that regular real-valued ones. Nevertheless, in what follows we will show that, when comparing our interval-based approach to real-valued processes that

Table 3: Generic branch-and-bound algorithm

Input: System of constraints $C = \{c_1, \ldots, c_k\}$, a search space D_0.
Output: A set Sol of interval solutions (boxes of size n, the number of variables)
Set Sol to empty
If $\forall i$, $0 \in F_i(D_0)$ then:
Store D_0 in some storage S[1]
While (S is not empty) do:
Take D out of S
If ($\forall i$, $0 \in F_i(D)$) then:
If (D is still too large[2]) then:
Split[3] D in D_1 and D_2
Store D_1 and D_2 in S
Else:
Store D in Sol
Return Sol

have to be repeated countless times, then the extra cost of interval computations is counterbalanced and our approach more computationally effective (as shown in Section 5).

3 Proper Orthogonal Decomposition

In this section, we study the statistical procedure of Principal Component Analysis (PCA), which uses orthogonal transformation to convert a set of observations of possibly correlated Random Variables into a set of linearly uncorrelated ones with the largest possible variance, named principal components. The number of principal components is less than or equal to the number of the original random variables.

Using the same procedure as in (PCA), it is possible to find a set of linearly independent vectors from a set of linearly dependent ones, whose spanned space is practically the same. This procedure is named Proper Orthogonal Decomposition (POD), which we also describe here.

3.1 Principal Component Analysis

When information from a data sample is collected, usually we take the maximum number of variables. However, if we take too many variables from a data sample, for instance 20 variables, we must consider $\binom{20}{2} = 190$ possible correlation coefficients. If you have 40 variables that number is increased to 780. Obviously, in this case it is difficult to visualize relationships between variables. Another problem that arises is the strong correlation that often occurs between variables: if we take too many variables (which generally happens when much is not known about data, or we are only interested in exploratory tests), it is normal that they are related or they measure the same thing under different viewpoints. For example, in medical studies, blood pressure at the heart's outlet and out of the lungs are strongly related.

Therefore, it is necessary to reduce the number of variables. It is important to highlight that the concept of major information is related to the greater variability of the data or variance. The greater the variability (variance) of the data, the more information this data has.

Studying the relationships that exist between p correlated variables (which commonly measure information) transforms the original set of variables in another new set of uncorrelated variables together (that has no repetition or redundancy on the information) called a set of principal components.

3.2 Principal Components

Let us consider a number of variables $X = (x_1, x_2, \cdots, x_n)$ describing a group of objects or individuals and to calculate, from them, a new set of variables $(y_1, y_2, \cdots, y_n)$ uncorrelated with each other, whose variances will decrease gradually.

Each y_j (where $j = 1, \ldots, n$) is a linear combination of the original variables $x_1, x_2, \ldots, x_n$, i.e

$$\begin{aligned} y_j &= v_{1j}x_1 + v_{2j}x_2 + \cdots + v_{pj}x_n \\ &= Xv_j, \end{aligned}$$

where $v_j^{\mathrm{T}} = (v_{1j}, v_{2j}, \cdots, v_{pj})$ is a constant vector.

To keep the orthogonality of the transformation, we impose $||v_j|| = 1$.

The first component v_1 is calculated so y_1 has the greatest variance subject to the constraint that $||v_1|| = 1$. The second principal component v_2 is calculated so that the variables y_1 and y_2 are uncorrelated. Similarly are chosen $y_1, y_2, \ldots, y_p$, uncorrelated with each other.

The full principal components decomposition of X can therefore be given as

$$Y = XV,$$

where V is a $p \times p$ matrix whose columns are the eigenvectors of $X^T X$.

The principal component decomposition of X can be expressed in terms of singular value decomposition of X. Given

$$X = U\Sigma V^T,$$

then we have

$$\begin{aligned} Y &= XV \\ &= U\Sigma V^T V \\ &= U\Sigma. \end{aligned}$$

In practice, we initiate computations with p variables and we are left with a number of much smaller components that collect a large percentage of the variability. For instance, we take r variables, where r is the minimum positive integer such that:

$$\frac{\sum_{i=1}^{r} \sigma_i}{\sum_{i=1}^{p} \sigma_i} > tol$$

where *tol* is an approximation of 1 by defect.

3.3 Proper Orthogonal Decomposition Method

Consider a parameterized static computational model described by large-scale linear system of discrete equations

$$R(x, \lambda) = 0. \tag{20}$$

Here we can see (20) as an input-output system, where λ is the input and the solution, $x(\lambda) \in \mathbb{R}^n$, is the output.

The idea behind this method is that, given a certain input, the solution $x(\lambda)$ of a system contains the behavior of the system [25]. Therefore, the set of outputs serves as a starting-point for POD. The outputs are called *snapshots* and these must be given or be computed first.

Assume the set of snapshots S and the solution $x(\lambda^*)$ of (9) for a particular λ^* is in the subspace spanned by S. We assume that the columns of S are highly correlated, so we can apply principal components analysis (PCA) to obtain an uncorrelated number of columns, see 3.1, and thus to reduce the size of linear system of equations.

Consider the SVD of S

$$S = U\Sigma V^T \tag{21}$$

and

$$T = V\Sigma^{-1}U^T. \tag{22}$$

Define

$$\begin{aligned} T_1 &= \sum_{i=1}^{k} v_i \sigma_i^{-1} u_i^T; \qquad T_2 = \sum_{i=k+1}^{n} v_i \sigma_i^{-1} u_i^T, \\ S_1 &= \sum_{i=1}^{k} u_i \sigma_i v_i^T; \qquad S_2 = \sum_{i=k+1}^{n} u_i \sigma_i v_i^T. \end{aligned} \tag{23}$$

Conditions given by (23) are a particular case of conditions given in (7). We conclude that we get a good approximation of (9) if $S_2\tilde{x}$ is sufficiently small ($\tilde{x} = T_2(x)$) or equivalently if $\sigma_i \approx 0$ for $k+1 \leq i \leq n$.

To obtain a basis of W we have the algorithm in Table 4.

Table 4: Computing a Proper Orthogonal Decomposition basis

In: Parameter λ's and *input-output system*
Out: Base of the subspace W
Solve the full-order model to several λ's.
For each λ, take one or more **snapshots**, which is the solution of (9) for some values of t, and store such snapshots in a matrix S. Compute the SVD of S: $[W, \Sigma, V] = svd(S)$.
Find k such that $\sigma = \frac{\sum_{i=1}^{k} \sigma_i}{\sum_{i=1}^{n} \sigma_i} > 0.99$.
Consider only the k first columns and redefine $W = W(:, [1:k])$.

Several problems have been solved by using this method [28]. As it has been said before, the POD is based on Principal Components Analysis. The reader who wants to read a little more about this can find a good source of information in [14].

4 Interval Proper Orthogonal Decomposition (I-POD)

In this section, we present our Interval POD approach to solving large nonlinear systems of equations in a reduced subspace. Let us first recall once again the problem that we are solving.

Given a parametric system of equations (also known as the Full Order Model):

$$R(x, \lambda) = 0, \qquad \lambda \in \mathbf{I} \tag{24}$$

where R can be either linear or nonlinear function $R : \mathbb{R}^n \to \mathbb{R}^n$, that might arise from the discretization of a set of partial differential equations and $\mathbf{I}$ is a fixed interval. The idea behind POD is to solve (24) for a sequence of values $\lambda_i \in \mathbf{I}$, i.e.,

$$\begin{aligned} R(x, \lambda_1) &= 0, \\ R(x, \lambda_2) &= 0, \\ &\vdots \quad\quad \vdots \\ R(x, \lambda_n) &= 0, \end{aligned} \tag{25}$$

where $\lambda_i \in \mathbf{I}$, for $i = 1, 2, \ldots, n$. The main idea of this method is based on the high correlation between solutions for such values λ_i, so PCA techniques can be applied to obtain a smaller number of columns uncorrelated with the greatest of accumulated variance.

In this work, we propose an interval version of POD. The original idea behind this new Interval POD is that we aim to reduce the amount of work in solving the Full Order Model for many different values of the input parameters (λ). Instead we suggest and experimented solving the Full Order Model once on the entire interval containing all desirable values of λ.

This slight change in concept (many processes solving for many different values of λ vs. one process solving for an entire interval instead) has consequences in our ability to solve the Full Order Model. Now that an interval is part of the problem we are bound to use interval-computation-based solving techniques and we found interval constraint solving techniques to be very practical to do so.

More specifically, we are now solving:

$$R(x, \mathbf{I}) = 0, \tag{26}$$

which is a nonlinear system of equations with explicit uncertainty in the shape of an interval.

We called this variation of POD the **Interval Proper Orthogonal Decomposition (I-POD)** method.

5 Numerical Results

In this section, we describe and report on preliminary experiments of our I-POD method on five problems. First, we present our results on the Burgers' equation, the Transport equation, the Lotka-Volterra problem, and the FitzHugh-Nagumo Model. Then we go on to studying an additional problem: the Bratu's equation, but due to its special features, we decided to explain the obtained results in a different section. For each of these experiments, we aim to assess the ability of I-POD to generate snapshots that yield a reduced basis of high-enough quality that the solution of the reduced-order model yields a very small error (w.r.t. FOM solution) in comparison to what a similar process using POD achieves. Our experiments were conducted using MATLAB R2012b (8.0.0.783) on a laptop with 1.7 GHz intel core i7 and 8GB of RAM.

5.1 Burgers' Equation

Consider the Burgers' equation:

$$\frac{\partial U(x,t)}{\partial t} + \frac{\partial f(U(x,t))}{\partial x} = g(x), \tag{27}$$

where U is the unknown conserved quantity (mass, density, heat etc.), $f(U) = 0.5U^2$ and in this example, $g(x) = 0.02\exp(0.02x)$. The initial and boundary conditions used with the above PDE are: $U(x;0) \equiv 1$; $U(0;t) = 4$, for all $x \in [0;100]$, and $t > 0$.

Below, in Tables 5 and 6, we describe the procedure to obtain the snapshots and the reduced basis in the POD method for the Burgers' equation. We will then compared it with **I-POD**.

Table 5: Computing a Proper Orthogonal Decomposition basis

Initialize an empty matrix where we will collect the snapshots: Snap = [], and an initial $\lambda = 3.5$.
`For i= 2:100`,

Solve:
$$\begin{cases} \dfrac{\partial U(x,t)}{\partial t} + \dfrac{\partial f(U(x,t))}{\partial x} = g(x), \\ g(x) = 0.02\exp(0.02x), \\ U(x;0) \equiv 1, \text{ for all } x \in [0;100], \\ U(0;t) = \lambda_i, \text{ for } t > 0. \end{cases} \tag{28}$$

Collect snapshots:

From $t_1, t_2, \ldots, t_n$, select a subsequence[4] $t_{i1}, t_{i2}, \ldots, t_{ip}$.

Add new columns to the snapshot matrix Snap = [Snap $U(x,t_{i1})$ $U(x,t_{i2})$ $\ldots U(x,t_{ip})$].

Update λ: $\lambda_i = \lambda_{i-1} + 0.01$

Apply the principal component analysis, (SVD). Snap $= W\Sigma V^T$.

Select from W the principal components with the greatest accumulated variance:

$\sigma = 0$.

`for k=1:n` compute:

$\sigma = \sigma + \frac{\sigma_k}{\sum_{j=1}^{n}\sigma_j}$

If $\sigma > Tol$, $0 < Tol < 1$, break.

Select the first k columns of W and redefine it. $W = W(:,[1,2,\ldots,k])$.

The new W will be the reduced basis to apply the POD method.

We applied both previous procedures, POD and I-POD, to solve (27) and we obtained the results reported in the Table 7. We observe that there is no significant difference between the traditional method (using POD) and the method we propose (using I-POD) w.r.t. (1) the dimension of the subspace, (2) the time it takes to solve the problem once we have identified the reduced basis, and (3) the relative error compared with the FOM solution. The major two advantages of our proposed method are:

- the computational time it requires to obtain the snapshots: Our approach requires 68.52% less time than the original one and the quality of the snapshots our method generates is comparable to that generated by POD as observed in the relative error; and

Table 6: Computing a Proper Orthogonal Decomposition basis using I-POD

Initialize a empty matrix where to collect the snapshots: Snap = [], and an initial $\lambda = 3.5$.

Solve:
$$\begin{cases} \dfrac{\partial U(x,t)}{\partial t} + \dfrac{\partial f(U(x,t))}{\partial x} = g(x), \\ g(x) = 0.02\exp(0.02x), \\ U(x;0) \equiv 1, \text{ for all } x \in [0;100], \\ U(0;t) = \mathbf{I}, \text{ for } t > 0. \end{cases} \quad (29)$$

The solution of (29) is an interval solution, i.e, for any $1 \leq x_0 \leq 100, 0 \leq t_0 \leq 50$, the value $U(x_0, t_0)$ is an interval. The infimum of such interval is defined $U_l(x_0, t_0)$ and $U_r(x_0, t_0)$ is the supremum. In that case, for all $1 \leq x \leq 100,\ 0 \leq t \leq 50$, $U(x,t) \in [U_l(x,t), U_r(x,t)]$, see Figure 3.

```
For i=1:100
```
Compute:
$$U(x,t) = (U_r(x,t) - U_l(x,t))(\lambda - 3.5) + U_l(x,t)$$
Collect snapshots:
From $t_1, t_2, \ldots, t_n$, select a subsequence[5] $t_{i1}, t_{i2}, \ldots, t_{ip}$.
Add new columns to the snapshot matrix Snap = [Snap $U(x,t_{i1})\ U(x,t_{i2})\ \ldots U(x,t_{ip})$].
Update λ: $\lambda_i = \lambda_{i-1} + 0.01$

Apply the principal component analysis, (SVD). Snap $= W\Sigma V^T$.
Select from W the principal components with the greatest accumulated variance:
$\sigma = 0$.
`for k=1:n` compute:
$$\sigma = \sigma + \frac{\sigma_k}{\sum_{j=1}^{n} \sigma_j}$$
If $\sigma > Tol$, $0 < Tol < 1$, break.
Select the first k columns of W and redefine it. $W = W(:, [1, 2, \ldots, k])$.
The new W will be the reduced basis to apply the POD method.

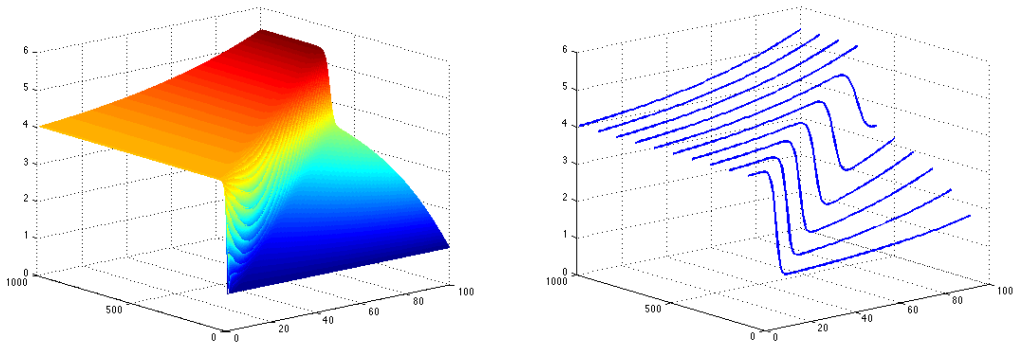


Figure 2: Solution of (28) for $\lambda = 4$, and some snapshots corresponding to this parameter

- the ability to handle uncertainty: the interval that contains λ, handled at once by IPOD, is similar to uncertainty and is handled without problems. Further experiments will aim to demonstrate that IPOD produces meaningful results also when there are other sources of uncertainty (beyond the interval for λ).

5.2 Transport Equation

The transport equation is a partial differential equation that models the concentration of a contaminant in the position x in at time t in a fluid that is flowing with velocity v in a thin straight tube whose ross section,

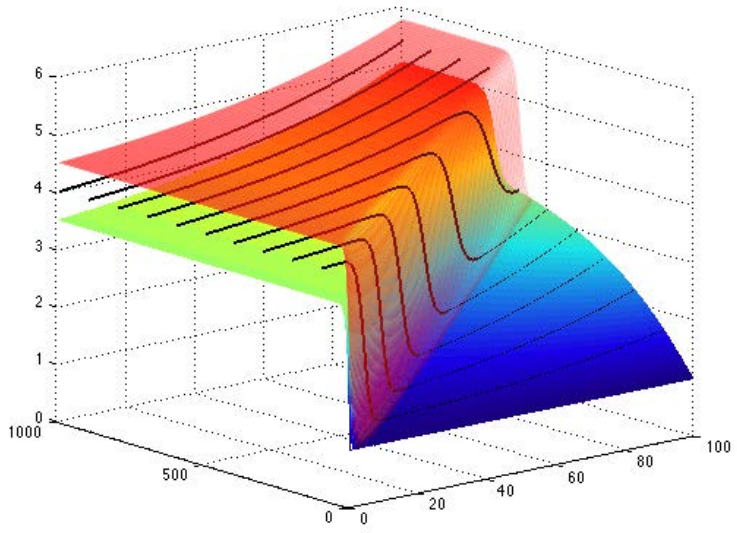


Figure 3: Infimum and supremum of the solution of (29), and some snapshots corresponding to $\lambda = 4$ are shown

Table 7: Comparing POD and I-POD methods in solving a particular example of Burgers' Equation

Method	FOM	POD	I-POD
Dimension	100	37	36
Time to solution	1.5 s	0.75 s	0.75 s
Relative error: $\|\|u_{fom} - u_{rom}\|\|/\|\|u_{fom}\|\|$	-	$4.85E-4$	$5.76E-4$

denoted by A is constant. Such concentration will be denoted by $U(x,t)$. if the function $U(x,t)$ and its partial derivatives of order one are continuous functions of x and t, and the fluid velocity v and the cross section of the tube, A, are constants, then the Transport Equation is reduced to:

$$\begin{aligned} &\frac{\partial U}{\partial t} + v\frac{\partial U}{\partial x} = 0 \\ &(x,t) \in \Omega \end{aligned} \tag{30}$$

Where Ω is a convex domain. In particular, we solve (30) with $U(x,t)$ subject to the following boundary and initial conditions:

$$U(0,t) = u(t) = -\sin(2\pi t) + \sin(\pi t) \tag{31}$$

$$U(x,0) = u(x) = \sin(2\pi x) + \sin(\pi x) \tag{32}$$

for all $t \in [0,1]$, and $x \in [0,1]$.

Using $v \in [0.5, 1.5]$ as the input parameter, we can proceed, similarly to how we did in the Burger Equation case, and compute, first, a basis using POD method, and later, using IPOD.

Comparative values are presented in Table 8:

In this experiment, we observed that even when the dimension of the subspace obtained with IPOD (76) is larger than the subspace obtained with POD (12) , once, both basis are known, solving the reduced problem from POD or I-POD takes about the same amount of time. The main achievement in this experiment is that we were able to handle a fair amount of uncertainty in the parameter v (we solved the FOM with $v = [0.5, 1.5]$ as part of our IPOD process). What helped us handle such uncertainty was the fact that when the Transport

Table 8: Comparing POD and I-POD methods in solving a particular example of transport equation

Method	FOM	POD	I-POD
Dimension	100	12	76
Time to solution	0.15 s	0.022 s	0.042 s
Relative error: $\|\|u_{fom} - u_{rom}\|\|/\|\|u_{fom}\|\|$	-	$7.97E-5$	$1.81E-5$

equation (30) is discretized, we obtain a linear system of equations, which is not too hard to handle with uncertainty.

In Figure 4, we can observe the plot of the solution of the transport equation for time-steps 20, 40, 60, 80. In green and red are respectively the infimum and the supremum of the interval containing the solution.

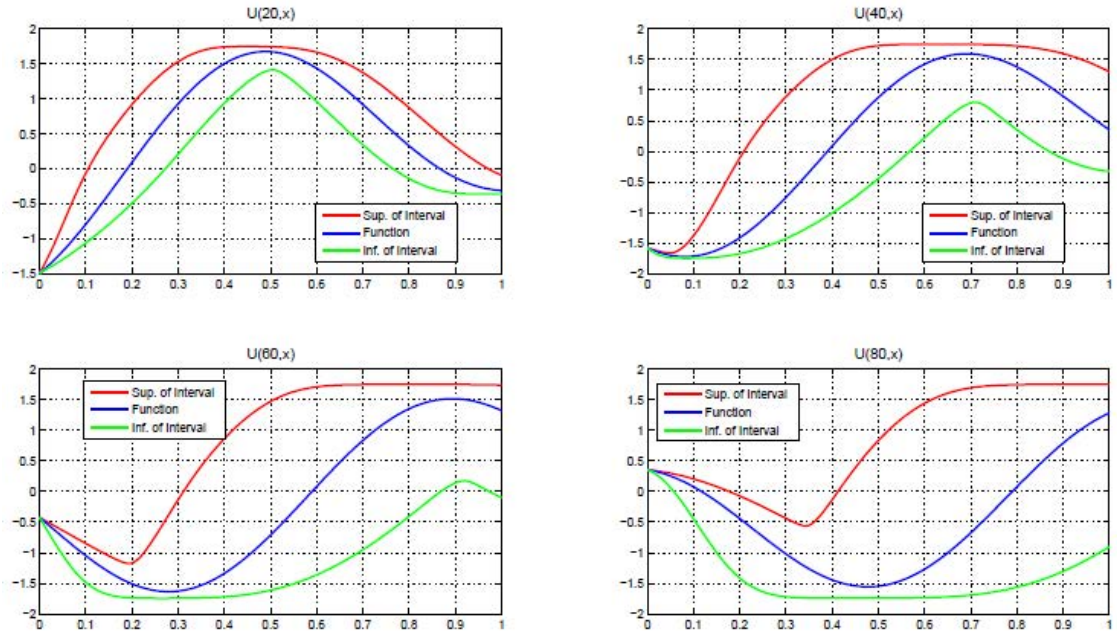


Figure 4: Solution of (30) for time-steps 20, 40, 60, 80, enclosed in the interval solution

5.3 Lotka-Volterra

Consider a particular case of the Lotka-Volterra problem, which involves a model of a predator-prey system.

$$\begin{cases} y_1' = \theta_1 y_1(1 - y_2), & y_1(0) = 1.2 \quad \theta_1 = 3, \\ y_2' = \theta_2 y_2(y_1 - 1), & y_2(0) = 1.1 \quad \theta_2 = 1. \end{cases} \tag{33}$$

y_1 and y_2 respectively represent the amount of preys and predators. In this particular example, the growth rate of the first species reflects the effect the second species has on the population of the first species (θ_1). Similarly, the growth rate of the second species reflects the effect the first species has on the population of the second species ($\theta 2$). The system was integrated from $t_0 = 0$ to $t_m = 10$. Numerical experiments were carried out with a constant step size $h = 0.1$. Ranges for the parameters $\theta_1 \in [2.95, 3.05]$ and $\theta_2 = [0.95, 1.05]$ were used as input. Comparative values are presented in Table 9:

In Figure 5, we observe the interval enclosure of (33) when $\theta_1 = [2.95, 3.05]$ and $\theta_2 = [0.95, 1.05]$.

The examples of the Burger's equation (27) and the Transport equation (30) were partial differential equations with uncertainty in one parameter. The number of unknowns was the same as the number of points in the discretization of the domain. Problem (33) is more challenging because it is a system of two nonlinear

Table 9: Comparing POD and I-POD methods in solving a particular example of Lotka-Volterra

Method	FOM	POD	I-POD
Dimension	200	3	3
Time to solution	0.0129 s	0.0746 s	0.0484 s
Relative error: $\|\|u_{fom} - u_{rom}\|\|/\|\|u_{fom}\|\|$	-	$4.56E-4$	$1.20E-3$

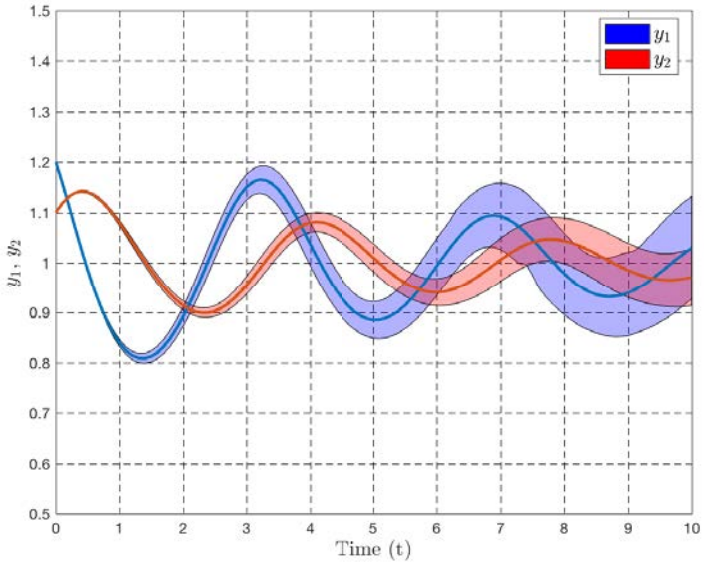


Figure 5: The interval enclosure of (33) for $\theta_1 = [2.95, 3.05]$ and $\theta_2 = [0.95, 1.05]$

partial differential equations, which means that the number of unknowns is twice as large as the number of nodes in the discretization. Also in this experiment, we show that we are able to handle uncertainty in two parameters. The results reported in Table 9 show that we can significantly reduce the size of the search space since we were able to go from dimension $n = 200$ to a reduced dimension $k = 3$, which constitutes a 98.5% contraction. Having uncertainty in two parameters did not yield a large loss of quality, since, given the reduction of the subspace, just one order of magnitude is lost.

5.4 The FitzHugh-Nagumo Model

The following nonlinear model is based on the classical FitzHugh-Nagumo oscillator. Let

$$f(v) = v(v - \alpha)(1 - v)$$

and let (v_{eq}, w_{eq}) be the equilibrium point of the nonlinear system. This system has been modified so that the equilibrium point coincides with the initial condition, i.e., $(v(0), w(0)) = (v_{eq}, w_{eq})$

$$\begin{cases} \dfrac{dv}{dt} &= f(v + v_{eq}) - f(v_{eq}) - w \\ \dfrac{dw}{dt} &= \varepsilon(v - \gamma w) \end{cases} \tag{34}$$

We will illustrate the behavior of the FitzHugh-Nagumo model using the following values for the parameters: $\alpha = 0.139$, $\varepsilon = 0.008$, $\gamma = 2.54$, $v_0 = v_{eq} = 0.15$, $w_0 = w_{eq} = -0.028$, the domain $t = [0, 10]$ was discretized using $\Delta_t = 0.1$.

Table 10: Comparing POD and I-POD methods in solving a particular example of the FitzHugh-Nagumo model

Method	FOM	POD	I-POD
Dimension	200	3	2
Time to solution	0.022 s	0.079 s	0.188 s
Relative error: $\|\|u_{fom} - u_{rom}\|\|/\|\|u_{fom}\|\|$	-	$6.28E-05$	0.0110

Let us consider the initial condition $v_0 = [0.1, 0.2]$ as the input parameter. Observe in Table 10 the comparative values, and in Figure 6 an enclosure of the solution when $v_0 = [0.1, 0.2]$.

Function f in the definition of FHN (34) is highly nonlinear. As a consequence, the nonlinear system of equations obtained when discretizing the domain is highly sensitive to overestimation when we use interval computations. In Table 10 we report our experimental results, which show that the performance of IPOD is affected by the highly nonlinear nature of the function we simulate. We observe that the time to solution of the ROM, even in a smaller space than POD, is larger. Similarly, the obtained accuracy (relative error to FOM) of the obtained solution is not nearly as good as that of POD. This is definitely an area of improvement, but overall, we observe that this problem does not naturally lend itself to POD (or IPOD for that matter) since even POD's time to solution is larger that that of FOM.

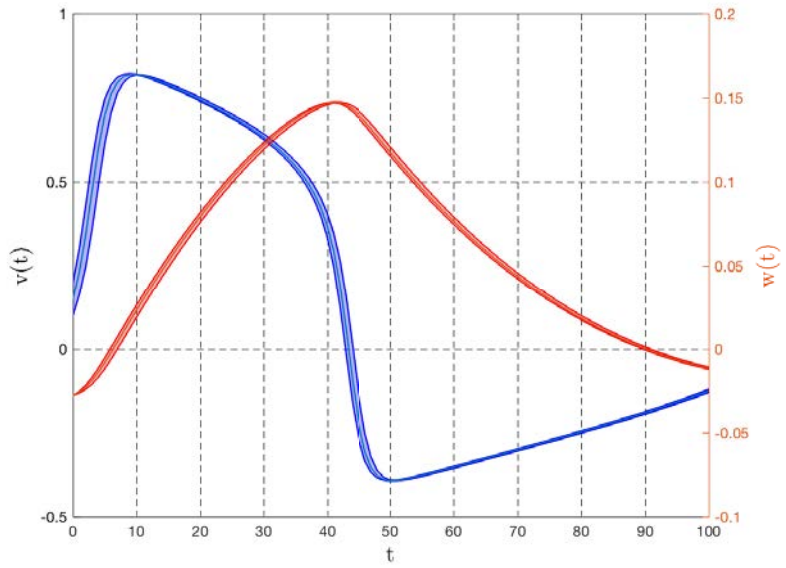


Figure 6: The interval enclosure of (34) for $v_0 = [0.1, 0.2]$

6 The Bratu Problem

In this section, we study the Bratu's problem, which is an the elliptic PDE:

$$\begin{aligned} \Delta u + re^u &= 0 \qquad \text{on } \Omega : \{(x,y) \in 0 \leq x \leq 1, 0 \leq y \leq 1\} \\ \text{with } u &= 0 \qquad \text{on } \partial\Omega. \end{aligned} \tag{35}$$

Studying Bratu's problem in more details is interesting because the existence and uniqueness of (35) depends on the parameter r. There actually exists a critical value r_*, called the **Frank-Kamdndtskii** value, such that for $r > r_*$, there does not exist any solution to the Bratu's problem, and two solutions exist for $0 < r < r_*$ [7]. Once r_* has been determined, the finite difference only gives the lower branch of

the solution [23]. Experimentally, it has been proven that for (35) the Frank-Kamdndtskii value is around $r_* \approx 6.78$ [6, 2]. Zero-Dirichlet solutions are used to model combustion reactions.

6.1 Existence and Uniqueness

We consider the discretization of the domain Ω, $u(x_i, y_j) = u_{i,j}$ with $1 \leq i, j \leq 30$ (36).

$$\begin{aligned} \frac{u_{i,j-1} - 2u_{ij} + u_{i,j+1}}{(\Delta y)^2} + \frac{u_{i-1,j} - 2u_{ij} + u_{i+1,j}}{(\Delta x)^2} + re^{u_{ij}} &= 0 \\ u_{0,j} = u_{31,j} = u_{i,0} = u_{i,31} &= 0 \end{aligned} \tag{36}$$

After substituting $r = [6.79, \infty]$ in (36) and using ICST to solve it. We can prove that (36) has no solution for $r > 6.79$. If $r \to 0$ then $u_{ij} \to 0$ for all $1 \leq i, j \leq 30$, we will focus in the solutions for $1 \leq r \leq 6.78$.

Now, let us consider the parameter $r = 1$, and apply ICST. In order to use ICST to solve the problem, we need to set the parameter as an interval $r = [1, 1]$. In that case we obtain two solutions for (36) see Figure 7.

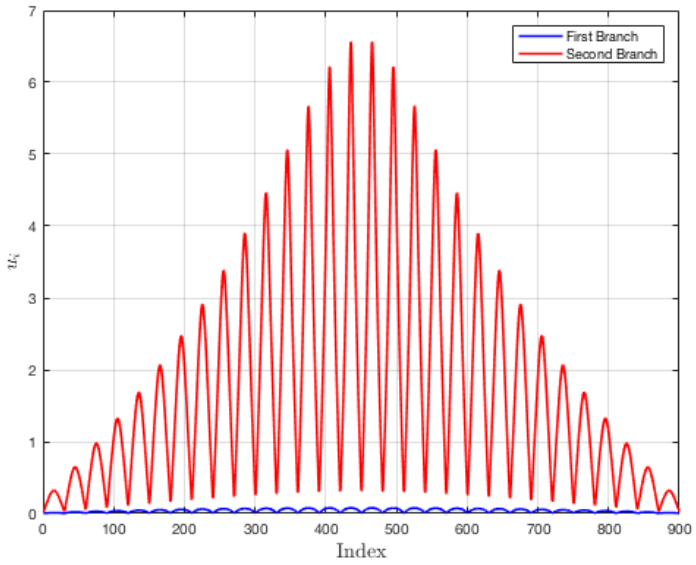


Figure 7: Two solutions of (36) for $r = [1, 1]$

6.2 Comparison POD and I-POD with Bratu's Problem

In order to perform a comparison between POD and I-POD, we first need to take 100 uniformly distributed values of parameter $r \in [1, 6.78]$ to compute the snapshots. For each r, the solution is column-wise sorted and stored in the snapshot matrix. Later, we use ICST to solve the same problem but now $r = [1, 6.78]$. Results are shown in Table 11.

Table 11: POD and I-POD methods in solving a particular example of the Bratu problem

Method	FOM	POD	I-POD
Time to compute the reduced basis	-	2.56 s	7,200 s
Dimension	900	4	4
Time to solution	66.4 ms	11.2 ms	14.4 ms
Relative error: $\|\|u_{fom} - u_{rom}\|\|/\|\|u_{fom}\|\|$	-	$9.94E-04$	0.0045

Figure 8 shows a solution's enclosure of (35) for $r = [1, 6.78]$.

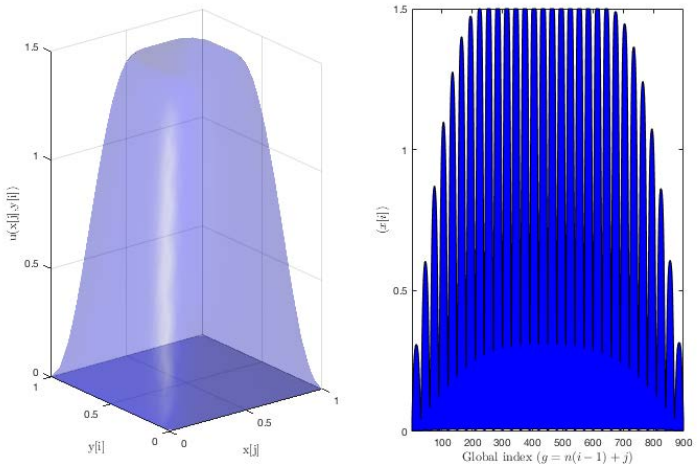


Figure 8: The interval enclosure of (35) for $r = [1, 6.78]$

6.3 ICST and MOR

We have seen how the time to the solution can been improved using MOR, but we also have seen that using MOR with the traditional methods, such as Newton-like methods, converges to only one solution, regardless of whether the problem has more than one solution. On the other hand, it has been proven that with ICST, we can obtain all the solutions of a problem, but it takes longer to identify all these solutions. Under such circumstances, it is natural to wonder: is it possible to benefit from the advantages of ICST of obtaining all the solutions of a problem and reducing the time using MOR?

In order to answer the previous question, we solved the problem (36), but, now, finding the solution in a subspace containing both solutions of the Bratu's equation. Let us define Φ a basis of such subspace of dimension 4. Equation (36) becomes in an over-determined nonlinear system of equations $F(\Phi p) = 0$. In this case, the number of unknowns is reduced, but the system still has the same number of equations (constraints). In Table 12, we have the reduced interval solutions for (35).

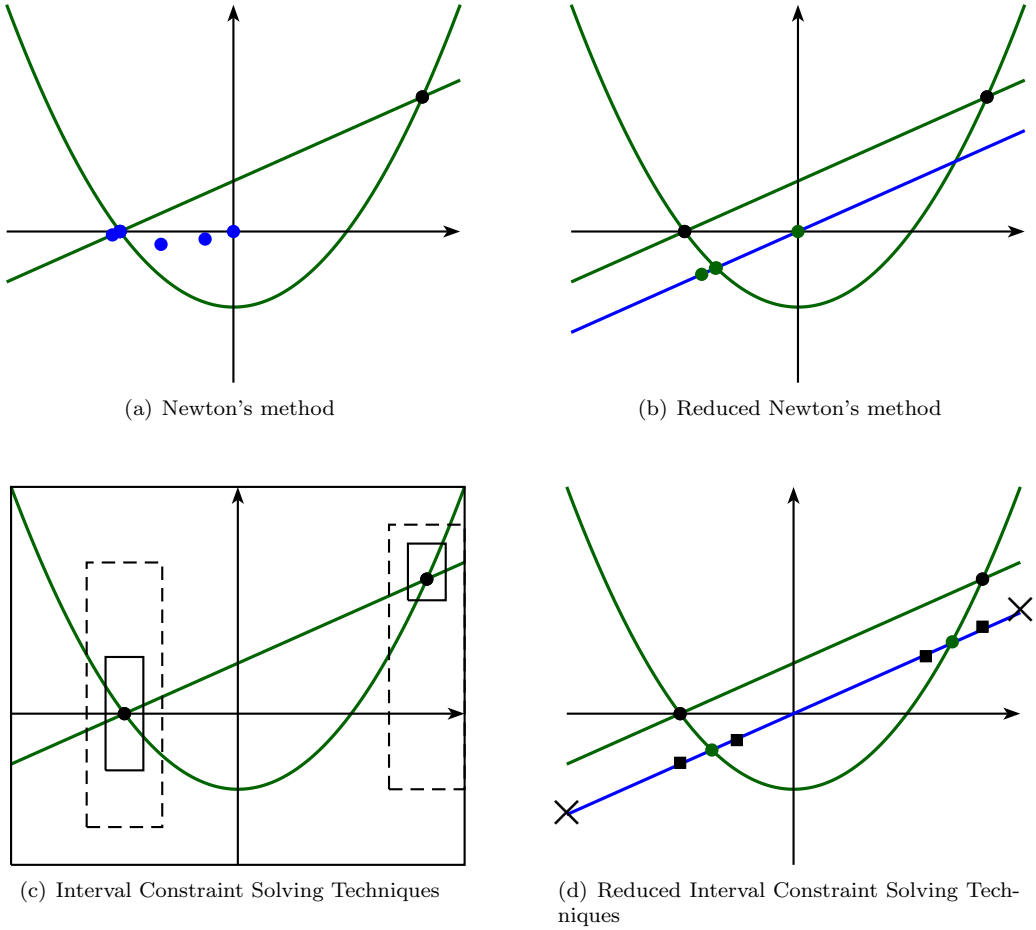
Table 12: Reduced interval solution for Bratu's problem

	Branch 1		Branch 2	
	Lower Bound	Upper Bound	Lower Bound	Upper Bound
P_1	-5.23478779E-06	4.57662756E-06	-1.00002058	-0.99998193
P_2	-1.00000152	-0.99999855	-1.00001805	-1.00000587
P_3	-1.00000110	-0.99999890	-2	-1.99995535
P_4	0.99999653	1.00000387	2.99994642	3.00000857

The time to the solution for FOM-ICST was 7200s, and the time for MOR-ICST was 188s, which represents a very significant improvement.

Before to close this section, let us illustrate the four methods studied here. In Figure 9, the four methods studied in this article are represented. A simple nonlinear system of equations is illustrated and the Newton's method is used to solve it Figure 9(a), observe how the different approximations of the Newton's method, the blue dots, converge to a solution of the system. The blue line in Figure 9(b) represents a reduced subspace where an approximation of the solution is sought. In this case, the different approximations converge to the intersection of the subspace with the quadratic function. This intersection is the best approximation to the solution. In both cases, either method converges to only one solution of the system. The method illustrated in Figure 9(c) shows how ICST work. Observe in this case how this method reduces and splits the initial box, which is the solid square enclosing the whole figure, until it encloses the two solutions of the system. Finally,

Figure 9(d) illustrates the Reduced ICST. The two symbols $\times$ in the extremes of the subspace represent the lower and upper bounds of the initial box. In this case the initial box is splitted in two intervals and then each subinterval is shrunk to enclose both solution approximations of the system.



(a) Newton's method

(b) Reduced Newton's method

(c) Interval Constraint Solving Techniques

(d) Reduced Interval Constraint Solving Techniques

Figure 9: Different methods to solve a nonlinear system of equations

7 Conclusions and Future Work

In this article, we proposed and described a novel Model-Order Reduction (MOR) approach that improves the well-known Proper Orthogonal Decomposition method (POD) by 1/ freeing traditional MOR techniques from snapshot identification, 2/ providing reliable solutions, and 3/ allowing to handle uncertainty. Our new approach is based on the use of Interval analysis and Interval Constraint Solving Techniques. We called this new method the Interval Proper Orthogonal Decomposition (I-POD). We tested I-POD on five nonlinear partial differential equations problems: Burgers' equation, the Transport equation, Lotka-Volterra problem, the FitzHugh-Nagumo Model, and the Bratu's Problem. We observed and reported promising performance of I-POD, when compared to POD.

From this preliminary work, we draw the following research activities and directions. First, we will keep challenging I-POD with problems that are even larger and more non-linear. We will study potential limits of uncertainty in the original model: how to quantify the amount of uncertainty that still allows to draw conclusions from simulations? We then plan to draw applications from IPOD: for instance, can we use it for prediction of future behavior based on observations? If so, how can we translate observations of the full-order model to values of the reduced-order model? How much more uncertainty does that translation bring?

Acknowledgments

This work was supported by Stanford's Army High-Performance Computing Research Center funded by the army Research Lab, and by the National Science Foundation award #0953339.

References

[1] Araya, I., Neveu, B., and G. Trombettoni, An interval extension based on occurrence grouping, *Computing*, vol.94, no.2, pp.173–188, 2012.

[2] Argaez, M., Florez, H.A., and O. Mendez, A model reduction for highly non-linear problems using wavelets and Gauss-Newton method, *Proceedings of NAFIPS*, 2016.

[3] Björck, A., *Numerical Methods for Least Squares Problems*, Society for Industrial and Applied Mathematics(SIAM), 1996.

[4] Caroa, S., Chablata, D., Goldsztejnb, A., Ishiic, D., and C. Jermannd, A branch and prune algorithm for the computation of generalized aspects of parallel robots, *Artificial Intelligence*, vol.211, pp.34–50, 2014.

[5] Cotlar, M., and R. Cignoli, *Introduction to Functional Analysis*, 1st Edition, North Holland Publishing Company and Elsevier Publishing Company, 1974.

[6] Florez, H.A., and M. Argaez, Applications and comparison of model-order reduction methods based on wavelets and POD, *Proceedings of NAFIPS*, 2016.

[7] Gelfand, I.M., Some problems in the theory of quasi-linear equations, *American Mathematical Society*, vol.29, pp.295–381, 1963.

[8] Granvilliers, L., and F. Benhamou, RealPaver: an interval solver using constraint satisfaction techniques, *ACM Transactions on Mathematical Software*, vol.32, no.1, pp.138–156 2006.

[9] Hailer, A., *Verification of Branch and Bound Algorithms Applied to Water Distribution Network Design*, Dissertation, Technische Universität Hamburg, HarburgLogos Verlag, Berlin, 2006.

[10] Hansen, E., and W. Walster, *Global Optimization using Interval Analysis*, 2nd Edition, Inc. New York, 2004.

[11] Hentenryck, V.P., McAllester, D., and D. Kapur, Solving polynomial systems using a branch and prune approach, *Siam Journal on Numerical Analysis*, vol.34, no.2, pp.797–827, 1997.

[12] Hoffman, K., and R. Kunze, *Linear Algebra*, Prentice-Hall, 1971.

[13] Jaffar, J., and M. Maher, Constraint logic programming: a survey, *Journal of Logic Programming*, vols.19-20, pp.503–581, 1994.

[14] Jolliffe, I.T., *Principal Component Analysis*, Springer-Verlag, 1986.

[15] Kearfott, R.B., Verified branch and bound for singular linear and nonlinear programs: an epsilon-inflation process, 2007 (Available from the author).

[16] Kelly, C.T., Solving nonlinear equations with Newton's method, *Society for Industrial and Applied Mathematics*, vol.46, no.4, pp.770–771, 2003.

[17] Kelly, C.T., Reduction of model order based on proper orthogonal decomposition for Lithium-Ion Battery Simulations, *Journal of the Electrochemical Society*, vol.156, pp.A154–A161, 2009.

[18] Kreinovich, V., Xiang, G., Starks, S.A., Longpré, L., Ceberio, M., Araiza, R., Beck, J., Kandathi, R., Nayak, A., Torres, R., and J.G. Hajagos, Towards combining probabilistic and interval uncertainty in engineering calculations: algorithms for computing statistics under interval uncertainty, and their computational complexity, *Reliable Computing*, vol.12, no.6, pp.471–501, 2006.

[19] Li, J., and Y. Chen, *Computational Partial Differential Equations Using MATLAB*, CRC Press, Las Vegas (NV), 2008.

[20] Mackworth, A.K., Consistency in networks of relations, *Artificial Intelligence*, vol.8, no.1, pp.99–118, 1977.

[21] Marquez, A., and J. Espinosa, Model reduction using proper orthogonal decomposition and predictive control of distributed reactor system, *Journal of Control Science and Engineering*, vol.2013, Article ID 763165, 2013.

[22] Moore, R.E., Kearfott, R.B., and M.J. Cloud, *Introduction to Interval Analysis*, 1st Edition, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2009.

[23] Odejide, S.A., and A.S. Aregbesola, A note on two dimensional Bratu problem, *Kragujevac Journal of Mathematics*, vol.29, pp.49–56, 2006.

[24] Rathinam, M., and A. Petzold, A new look at proper orthogonal decomposition, *SIAM Journal on Numerical Analysis*, vol.41, pp.1893–1925, 2003.

[25] Schilders, W.H., and H.A. Vorst, *Model Order Reduction: Theory, Research Aspects and Applications*, Springer Science & Business Media, 2008.

[26] Sharan, M., and A. Pradhan, A numerical solution of Burgers' Equation based on multigrid method, *International Journal of Advancements in Electronics and Electrical Engineering*, vol.2, 2013.

[27] White, J., A trajectory piecewise-linear approach to model order reduction and fast simulation of nonlinear circuits and micromachined devices, *International Conference on Computer Aided Design*, 2013.

[28] Willcox, K., and J. Peraire, Balanced model reduction via the proper orthogonal decomposition, *AIAA Journal*, vol.40, no.11, pp.2323–2330, 2002.